Composing Diverse Policies for Temporally Extended Tasks

Composing Diverse Policies for Temporally Extended Tasks

Daniel Angelov, Yordan Hristov, Michael Burke and Subramanian Ramamoorthy

***Abstract*—Robot control policies for temporally extended and sequenced tasks are often characterized by discontinuous switches between different local dynamics. These change-points are often exploited in hierarchical motion planning to build approximate models and to facilitate the design of local, region-specific controllers. However, it becomes combinatorially challenging to implement such a pipeline for complex temporally extended tasks, especially when the sub-controllers work on different information streams, time scales and action spaces. In this paper, we introduce a method that can automatically compose diverse policies comprising motion planning trajectories, dynamic motion primitives and neural network controllers. We introduce a global goal scoring estimator that uses local, per-motion primitive dynamics models and corresponding activation state-space sets to sequence diverse policies in a locally optimal fashion. We use expert demonstrations to convert what is typically viewed as a gradient-based learning process into a planning process without explicitly specifying pre- and post-conditions. We first illustrate the proposed framework using an MDP benchmark to showcase robustness to action and model dynamics mismatch, and then with a particularly complex physical gear assembly task, solved on a PR2 robot. We show that the proposed approach successfully discovers the optimal sequence of controllers and solves both tasks efficiently.**

***Index Terms*—Motion and Path Planning; Learning and Adaptive Systems; Learning from Demonstration**

I. Introduction

FOR robots to work in the wild, they need to be able to perform a variety of consecutive tasks that might require vastly different skills. Each individual skill could be partitioned and optimized outside of this complex system, and is potentially constructed using a number of diverse methods or control strategies, such as motion planning approaches for reaching, contact aware grasping, picking and placing, or through the use of end-to-end neural network based controllers.

This paper was recommended for publication by Editor Nancy Amato upon evaluation of the Associate Editor and Reviewers' comments. This research is supported by the Engineering and Physical Sciences Research Council (EPSRC), as part of the CDT in Robotics and Autonomous Systems at Heriot-Watt University and The University of Edinburgh. Grant reference EP/L016834/1., and by an Alan Turing Institute sponsored project on Safe AI for Surgical Assistance.

D. Angelov, Y. Hristov, M. Burke, S. Ramamoorthy are with Institute of Perception, Action and Behaviour (IPAB), School of Informatics, The University of Edinburgh, EH8 9AB, UK; {d.angelov, yordan.hristov, michael.burke, s.ramamoorthy}@ed.ac.uk

In many practical applications, we wish to combine a diversity of such controllers to solve complex tasks. This typically requires that controllers share a common domain representation and a notion of progress to sequence these. For instance, the problem of assembly, as shown in Figure 1, can be partitioned by first picking up a mechanical part, then using motion planning and trajectory control to move this in close proximity to an assembly, before the subsequent use of a variety of wiggle policies to fit the parts together, as shown by [1]. Alternatively, the policy could be trained in an end-to-end fashion with a neural network, but one may find this difficult for extended tasks with sparse rewards, such as in Figure 1. In the interest of sample efficiency and tractability, such end-to-end learning could be warm-started by using samples from a motion planner, which provides information on how to bring the two pieces together and concentrates effort on learning an alignment policy, as in [2]. Additionally, the completion of these independent sub-tasks can be viewed as a global metric of progress.

We propose a hybrid hierarchical control strategy that allows for the use of diverse sets of sub-controllers, consisting of commonly used goal-directed motion planning techniques, other strategies such as wiggle, slide and push-against [3] that are so elegantly used in human manipulation, as well as deep neural network based policies that are represented very differently from their sampling-based motion planning counterparts.

Thus, we tackle a key challenge associated with existing motion primitive scheduling approaches, which typically assume that a common representation is used by all sub-controllers. We make use of the fact that controllers tend to have a dynamic model of the active part of their state space - either an analytical or a learned model, and further estimate how close each state is to completing the overall task using a novel goal scoring estimator. This allows the hierarchical controller to model the outcome of using any of the available sub-controllers and then determine which of these would bring the world state closest to achieving the desired solution – in the spirit of model predictive control.

As in the work of [4] on sequencing funnels and [5] on LQR-Trees, the scheduled controllers for sub-regions of the state space can be optimized in our framework, allowing for compositional task completion, but importantly, also for additional diversity of the controller set.

Value function approximation techniques used in the reinforcement learning community [6] can be considered similar to the proposed progress estimator, but only model the expected reward and require the actions to be in the same state space.

(a) Gear Pick Up (b) Move Gear (c) Gear Insertion

Fig. 1: Robot setup for the gear assembly task. The robot needs to pick up a gear by leveraging the surface of the table, slide it up to an edge, grasp and move it in a collision free manner to the other hand, before inserting the gear onto the base plate.

We attempt to remedy this oversight, by allowing for a diversity of action and state spaces, and by modelling global progress at a local controller level.

This paper makes the following contributions:

- We use a **Goal Score estimator** to sequence a set of policies to solve a task. This estimator is trained using expert demonstrations to evaluate the current and future state of the plan and helps to transform the hierarchical learning problem into a **planning** problem.
- We provide a method for composing **diverse** policies that work with different input information, or decompose the action in either joint or end-effector space and work at different operational frequencies to solve a high level task.

We first evaluate the use of the controller dynamics and the goal metric to compose policies in a hybrid controller on an MDP benchmark problem to evaluate robustness to action and model dynamics noise. Next, we apply this approach to a physical gear assembly task performed by the PR2 robot, making use of both motion planning and visual neural network policies (Figure. 1).

II. Related Work

Robotics: Compositionality is a key paradigm for robot control, which methods of composing controllers of a single type like [4], [5], [7] aim to exploit. These techniques rely on partitioning a state space into smaller overlapping operating regions and tuning sub-controllers (feedback or LQR) for operation in these regions. Unfortunately, these methods often fail to consider the fact that different tasks may require different controller sequences, and the scheduling of control laws in work on compositionality is often underemphasized. Inspired by this capability and the *funnels* framework [8][1] this work provides a Model Predictive Control (MPC) [9] framework for compositional sequencing where controllers can be of different types and operate using different state spaces.

[1]Regions of robustness arising from the dynamics and control applied in a sub-region of the control space.

The ability to act on different state-spaces and action sets is particularly important, as the sub-policies required to complete a temporally extended task can be highly variable. For example, sub-problems such as grasping and pushing have been addressed and investigated at least since the 1980s, and these could be encapsulated into operation as motion primitives [3]. Using a diverse set of policies allows for the selection of controllers that best fit the working domain - for example [10] highlights that compliance may be needed when movement and sensing reaches the perception noise boundary, [11] advocate using non-prehensile grasps for manipulation of objects and [12] explore manipulation strategies that allow for caging of objects, such that these can be re-grasped stably in a subsequent stage. Alternatively these motion planning strategies can be formulated using stable nonlinear attractor systems as in DMPs [13], [14] or as DeepDMPs [15]. We aim to create a hybrid control framework that allows the use of these diverse motion planning controllers, alongside neural network policies to solve long-sequence tasks.

Learning from Demonstration: To expedite the learning process, it is common to provide demonstrated example solution trajectories to a problem. Methods like Behaviour Cloning (BC) allow for simple visuomotor policies to be learned end-to-end [16], or to be extended to learn safe policies [17], extract preferences [18] or to learn mappings for the perception and kinematic differences [19]. Alternatively, they can be used to calculate the relative value of each state through inverse reinforcement learning and to create a hierarchical formulation for control [20]. As explained in [21], there are limitations to BC in terms of number of demonstrations, generalization, and the challenge of modeling complex scenarios. However, we use these full task demonstrations as a means for estimating the distance to a desired goal state, which is arguably a simpler task than learning an entire policy. Additionally, by allowing different controller representations, we do not need to re-represent one control law in alternative approximate forms.

Reinforcement Learning: In the reinforcement learning (RL) literature the concept of options has parallels to our work, as each policy can be viewed as a controller with the initiation

set as its domain. Our method lies between learning policies over options as in [22], and computing solutions using learning from demonstration by inverse reinforcement learning [23].

The options framework [24], [25] provides a formal means to work with hierarchically structured sequences of decisions made by a set of RL controllers. Temporal abstractions have been extensively investigated [26], [27], [28], [29], [24], and it is clear that hierarchical structure helps to simplify control, allows an observer to disambiguate the different states of the agent, and encapsulates a control policy and termination of the policy within a subset of the state space of the problem. This split in the state space allows us to verify the individual controller within the domain of operation [30], [31], deliberate about the cost of an option and increases interpretability [32]. Our work can be viewed as using a planner as a hierarchical policy in the options framework, which is made possible through the incorporation of a goal-scoring progress function learned from demonstration.

In a similar manner, [33] showed how planning can be incorporated into action selection when future states can be evaluated. Our method borrows this view of temporally abstracting trajectories and extends it by applying a dynamics model for each of the options, allowing an agent to assess its states and incorporate foresight [34] in its actions.

The work of [35] highlights that including a dense reward indeed increases the overall performance of the agent. Instead of using a predetermined dense function, we learn a Goal Scoring estimator from the demonstrations. As shown in [2] naively tuning and shaping a reward function may result in sub-optimal solutions using base actions. Furthermore, our planner selects an already learned controller and thus avoids converging to sub-optimal behaviours.

As highlighted by Sunderhauf [36], there are limits of the use of RL in robotics. By leveraging strategies from both RL and control communities, this work aims to increase the scope of problems that can be tackled in robotics.

III. Method

Our framework defines a hierarchical controller over the set of pre-existing controllers. Each policy uses its dynamic model to propagate the current state to a future state conditioned on its control law. The Goal Scoring Estimator, learned over expert demonstrations, evaluates those future states and selects a controller that brings the system closest to the desired configuration.

Formally, assume the existence of a learned set of controllers $\mathcal{C} = \{c_1, c_2, ..., c_N\}$ including those learnt from experience in previously solved problems. Using notation similar to the RL options framework [24], each controller c_ω is independently defined by a control law $\pi_\omega(s) \rightarrow a$, $s \in \mathcal{S}_\omega$, action $a \in \mathcal{A}_\omega$, a working domain $\mathcal{I}_\omega, \mathcal{I}_\omega \subseteq \mathcal{S}_\omega$ where the controller can be started, and a termination criterion β_ω. We rely on a forward dynamics model $s_{t+1} \sim \mathcal{D}_\omega(s_t, a_t)$, which is a stochastic mapping, and a Goal Scoring metric $g \sim \mathcal{G}_{K_j}(s_t),\ 0 \leq g \leq 1$, that estimates the progress of the state s_t with respect to a desired world configuration. We assume $\mathcal{G}_{K_j}$ to change monotonically through the demonstrated trajectories. The different controllers can work on different state spaces $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, .., \mathcal{S}_N\}$ as long as there exists a space $\mathcal{S}^*$, such that $\mathcal{S}_i \subseteq \mathcal{S}^*$. This means there exists a higher or equal order state space, which maps the controller space of operation to regions of $\mathcal{S}^*$.

This work constructs a hybrid hierarchical controller $\pi_\Omega(\omega_t|s_t)$ that can choose the next controller c_{ω_t} that needs to be executed to bring a learned latent state s_t to some desired s_{final}. It uses the forward dynamics model $\mathcal{D}_\omega$ in an n-step Model Predictive Control (MPC) look-ahead, using a Goal Scoring metric $\mathcal{G}_K$ that evaluates how close s_{t+n} is to s_{final}.

As shown in Fig. 2, in this work, we use a variational autoencoder to learn a latent state s_t from image observations. We assume that each controller in the library has an associated forward dynamics model, trained to predict the next latent state, s_{t+1}. This provides us with an implicit mapping between states, and allows us to render an image of an expected scene for each controller that is applied. This scene prediction is then used by the goal score metric to evaluate the effect of choosing each controller and to select the most appropriate controller to be used at a given time step. In effect, this means that controllers act on the appropriate state components, but the underlying state representation used for controller selection is conditioned on image observations. Conditioning on images is feasible, as the robot head camera provides an overhead view of the entire workspace. While it may be possible to learn a shared state representation or mapping between states, this can be challenging (e.g. mapping from joint angles to images is extremely hard), while learning to predict the next latent state is a much easier task. Each of the framework components is described below.

A. Goal Score Evaluation

The key component of the proposed framework is the ability to evaluate how well a particular state s maps to parts of a demonstrated expert trajectory. This allows us to estimate the temporal distance of that state to the end of the demonstration (see Figure 2). In a similar manner to [37], who use adjacency of frames as positive and negative examples, we leverage the temporal sequence of the demonstration as a measure of task completeness.

We capture demonstrations of the global task (in its entirety) to use as a weak supervision for learning a goal scoring network that allows us to map a state to a progress estimation value $g \sim \mathcal{G}(s_t)$ for a given task. To build the Goal Scoring models, we use a convolutional network head with a Mixture Density Network (MDN) tail to encode the different goal representations based on image observations. The network predicts a distribution over the proximity of the current state to the desired goal state.

The first observation of a demonstration can be viewed as score 0 – far away from the goal state, whereas the final observation as score 1.0 – a target representation of the world. Even though there may not be a one-to-one mapping between the values within several demonstrations, we rely on the variability in their lengths being encoded within the different modes of the MDN of the Goal Scoring Model.

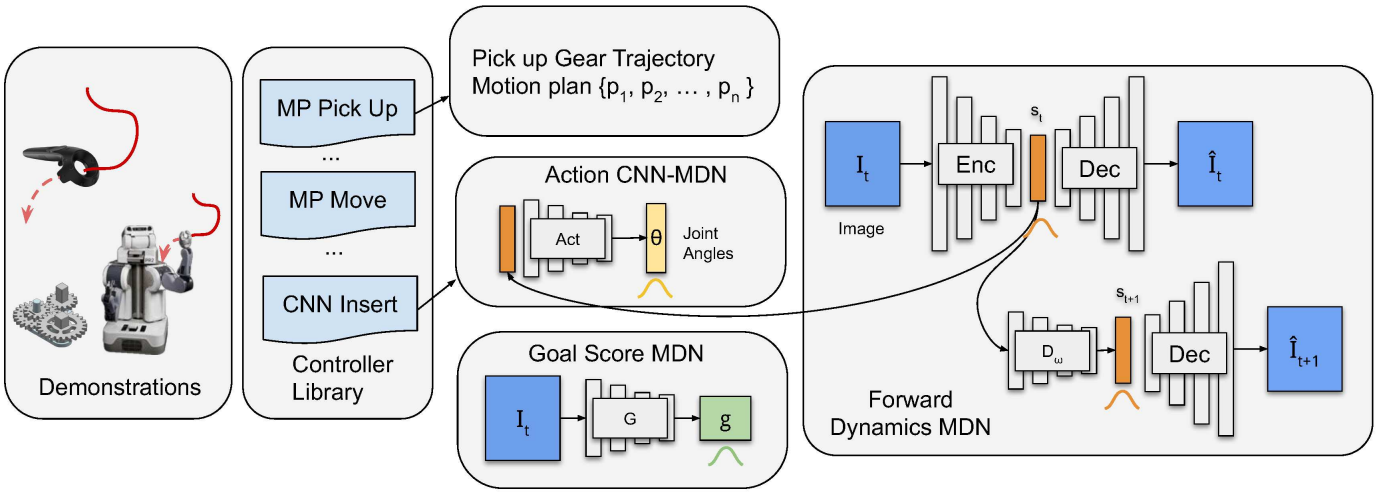


Fig. 2: Demonstrations were performed by using an HTC Vive controller that directly teleoperates the end-effector of the PR2 robot at 20 Hz. In the Gear Assembly Task, our controller library includes motion planning (MP) primitives (operating on joint angles) for picking up or moving and a convolutional neural network (CNN) for inserting the gear (operating on images). The MP primitives produce a trajectory for executing a task. The CNN policy takes a latent representation of the image state and generates a distribution over the target joint angles of the robot. The Forward Dynamics models use a VAE representation alongside an s_{t+1} dynamics prediction network that uses the same decoder. The Goal Score Estimator network takes in an image and produces a distribution over how well this image maps to a particular point in the demonstrations.

B. Controller Selection

At a particular point at state $s_t, s_t \in S^*$ when c_ω is active, we can compute the goodness of following the current controller given these conditions up to a particular time horizon. The action given by the policy is $a_t = \pi_\omega(\hat{s_t}), \hat{s} \in S_\omega$, and following the dynamics model we can write that:

$$s_{t+1} = \mathcal{D}_\omega(s_t, a_t) = \mathcal{D}_\omega(s_t, \pi_\omega(\hat{s_t})). \tag{1}$$

As the dynamics model is conditioned on the controller c_ω, we can simplify to $s_{t+1} = \mathcal{D}_\omega(s_t)$. Chaining this for n steps into the future we obtain

$$s_{t+n} = \mathcal{D}_\omega \circ \mathcal{D}_\omega \circ \cdots \circ \mathcal{D}_\omega(s_t) = \mathcal{D}_\omega^n(s_t). \tag{2}$$

We can evaluate this future state as $g_{t+n} = \mathcal{G} \circ \mathcal{D}_\omega^n(s_t)$. Thus, the hierarchical controller over controllers can be sequentially optimized,

$$\pi_\Omega(\omega_t|s_t) = \arg\max_\omega \left(\mathbb{E}\left[\mathbf{1}_{\mathcal{I}_\omega}(s_t) \cdot \mathcal{G} \circ \mathcal{D}_\omega^n(s_t)\right]\right) \tag{3}$$

This chooses the controller that is within the operation domain for the current state and delivers the largest goal score estimate after n steps. After choosing and evaluating the optimal π_Ω with respect to the above criterion, another controller can be selected at the next time step, with repetition until the goal is reached.

C. Controller Dynamics Modelling

The dynamics of each controller is modelled individually only within its operational domain. This simplifies the complexity the dynamics model has to learn and thus requires less data. Here, we learn a neural dynamics model for each controller that predicts the latent state configuration s_{t+1} from s_t, as in [38]. The architecture shown in Fig. 2 is based on a VAE encoding, but includes an additional dynamics network, which predicts the next latent state if a given controller were applied. The same decoder is used to force the two representations not to diverge.

A diverse dynamics network can be used as a prior for each controller [39] and the execution of the controllers themselves can be used to build an individual model using the image state space if it is not provided internally.

IV. Experimental Setup

We perform two sets of experiments to investigate the efficacy of the structured hierarchical policy by performing MPC future predictions at each step on a simulated MDP problem and on a much more complex physical gear assembly task on the PR2 robot.

A. Simulated MDP

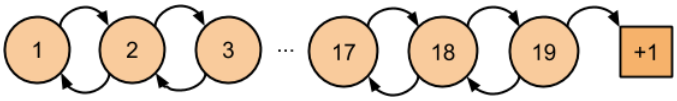


Fig. 3: The 19-state MDP problem. The action space of the MDP is to move "left" or "right". The goal of the MDP problem is to reach past state 19 and obtain the +1 reward, which is equivalent to a termination state 20.

In the first experiment, we use the standard 19-state random walk task as defined in [40] and shown in Figure 3 to illustrate concepts in a simple sequential decision making task. The goal of the agent is to reach past the 19^{th} state and obtain the $+1$ reward. The action space of the agent is to go "left" or "right", moving the agent to an increasing or decreasing

state. There also exist 5 controllers defined as in Section III, with the following policies: (1-3) policies that go "right" with a different termination probabilities $\beta = \{0.9, 0.5, 0.2\}$; (4) random action; (5) policy with action to go "'left" with $\beta = 0.5$. We assume that there exists a noisy dynamics model $\mathcal{D}_\omega$ and the goal evaluation model $\mathcal{G}_{MDP}$, which has the probability of falsely predicting the current state or its value of 0.2.

Further, we expand the MDP to be of size 100 and evaluate how sensitive the performance of the model is in regards to noise in the Goal Scoring evaluator and in each of the dynamics models.

B. Gear Assembly

In this task the PR2 robot needs to assemble the first part of the Siemens Challenge[2], which involves grasping a compound gear from a table, and placing it on a peg module held in the other hand of the robot. We record expert demonstrations of the task being performed, and assume access to a set of controllers that (1) picks up the gear from the table; (2) moves the left PR2 arm in proximity to the other arm; (3) inserts the gear on the peg module. Policy (1, 2) rely exclusively on scripted path planning techniques and work using discrete time steps, while (3) is learned entirely with a neural network. Controllers (1, 2) share a common state space of the robot's joint angles, whereas (3) works directly on the visual pixel input from the robot's head camera.

The visual neural policy, shown in Figure 2, performs imitation learning by using behaviour cloning of the 50 teleoperated demonstrations. This is trained until convergence or 100 epochs using different encoder heads - small convolutional network, ResNet-50, -101. The expert-illustrated trajectories were performed using a HTC Vive controller teleoperating the PR2 robot and the process took less than 1h wall time. The action generation part of the network is an MDN that predicts a distribution of the next time step joint angles θ, which are set as the internal PID targets for the robot 7-DOF arm.

The dynamics model for each controller is learned independently and is represented with a Forward Dynamics MDN, learned from forward rollouts of the policy network. The Goal Score estimator is learned on an additional 5 rollouts of the full gear assembly task and operates on the latent space of the particular policy. Throughout all of the experiments we use the Adam optimizer with a weight decay rate of $1e^{-}6$, batch size of 120, train for 200 epoch and the MDN uses 24 Gaussian mixtures. We show the performance of this model with several video streams from different cameras on the robot (head, left and right forearm cameras).

Additionally, we compare the performance of the scripted Motion Planning method (using RRT Connect [41]), Dynamic Motion Primitives (learned from the MPs) and the Visual Neural Policy on each subtask, as well as using the full sequence under the different controllers as a baseline.

[2]The challenge is at https://new.siemens.com/us/en/company/fairs-event

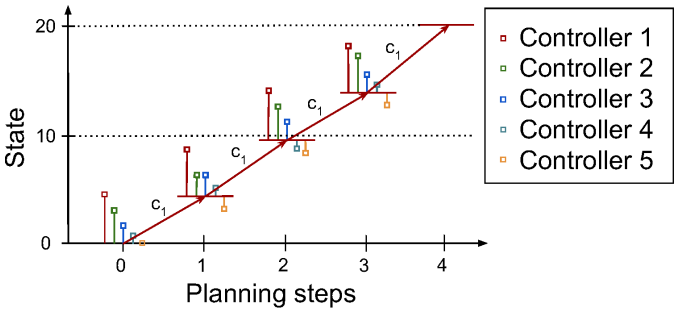


Fig. 4: MDP solution. At timestep 0, a rollout of the 5 controllers is performed with the dynamics model. The expected resulting state is marked using vertical bars. The best performing controller is used within the environment to obtain the next state - the red line at state 5 and planning step 1. This process is iterated until a desired state is reached.

V. Experimental Results

We demonstrate the viability of composing diverse policies by using the controller dynamics as a method for choosing a satisfactory policy. The dynamics can be learned independently of the task, and can be used to solve a downstream task.

Simulated MDP This problem illustrates the feasibility of using our architecture as a planning method. Figure 4 shows that the agent reaches the optimal state in just 4 planning steps, where each planning step is a rollout of a controller. The predicted state under the specified time horizon is illustrated at each step for the different controller options. This naturally suggests the use of the policy π_1 that outperforms the alternatives (π_1 reaches state 6, π_2 - state 4, π_2 - state 3, π_3 - state 1, π_4 - state 1, π_5 - state 0). Even though the predicted state differs from the true rollout of the policy, it allows the hierarchical controller to use the controller that would progress the state the furthest. The execution of some controllers (i.e. c_5 in planning steps 1, 2, 3) reverts the state of the world to a less desirable one. By using the forward dynamics, we can avoid sampling these undesirable controllers.

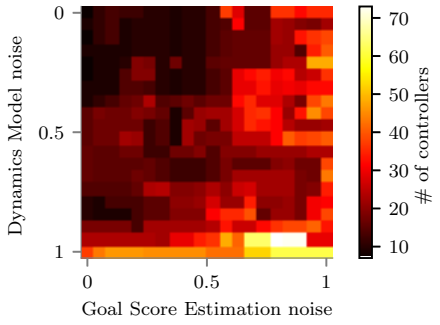
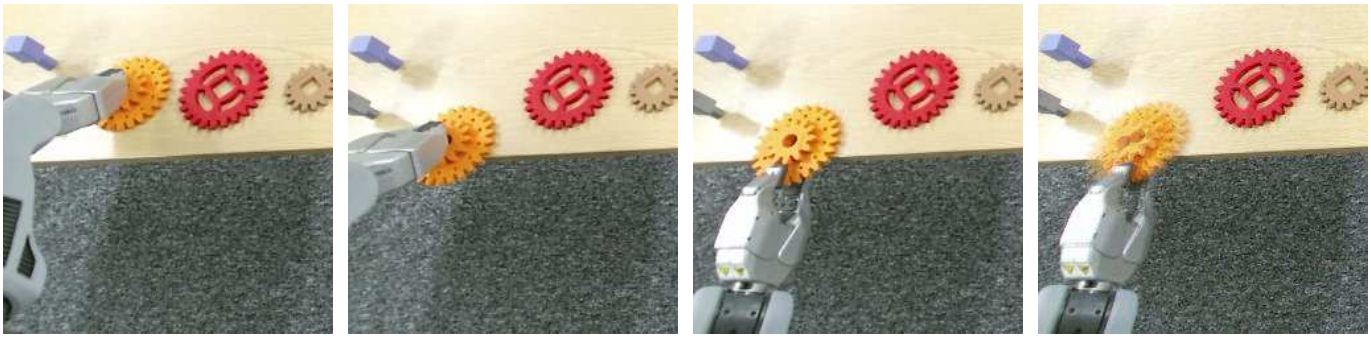


Fig. 5: Sensitivity to noise in the dynamics model and the Goal Score Estimator for a world of size 100. The heatmap illustrates the number of controllers that were used in order to reach the target with a lower number - top left - being optimal. The number of controllers varies between the optimal 8 and 72.



(a) Contact gear (b) Slide along table (c) Re-grasp stably (d) Grasp variability

Fig. 6: Images **(a-c)** illustrate key frames of the pick up policy that involves making physical contact with the gear, sliding it along the table surface to an edge and grasping it firmly in the new position. **(d)** A visual overlay of 3 random pickup attempts. The difference in grasp position relative to the gear is comparable to the inner diameter and is a byproduct of the stochasticity in the sliding and grasping action. This does not hinder the performance of the CNN in the full task.

In order to investigate the robustness and convergence properties of our method, we introduce noise within the system, while expanding the MDP to be of size 100 and maintaining the same 5 controllers as above. We can see in Figure. 5 how the number of controllers required to reach the target location varies at different noise levels. When we observe low amounts of noise, the performance remains stable and requires activating any of these controllers a total of less than 20 times (top-left part of the heatmap). The expected optimal number of controller activations based on policy 1 is 12 (black region of the heatmap). As the noise in both the dynamics model and the Goal Score Estimation increases, we observe a degradation and the selection of more sub-optimal controllers. The model is more sensitive to noise in the Goal Score Estimator than when the dynamics of the controllers make errors in their predictions. Despite this, the method converges to the optimal state.

It is interesting to note that the method uses close to or optimal number of controller activations in cases where multiple policies would drive the world in a progressive state, highlighting that the goal score metric is capable of choosing longer horizon controllers due to the MPC look ahead.

Gear Assembly We build the library of controllers for the task - picking up a gear (Figure 6), moving it close to the base of the assembly and inserting the gear on the base plate (Figure. 7). A motion planning control method was used to perform different tasks. Those demonstrations were used to build the DMP model, using the ROS-DMP module, which is based on [14]. The Convolutional Neural Network policy was trained using 50 tele-operated demonstrations covering a wide variety of initialization cases for each specific task. We did not observe any task performance changes between the small Convolutional or the ResNet-50,-101 head and therefore relied on the simple architecture. Other tasks may benefit from deeper or more complex models (such as [42], [43], [44]), but integration within the method would remain the same.

Table. I shows the performance of the different controllers on different tasks. The MP and DMP models exhibit stable performance in contact based tasks, but fail where the initial conditions differ – in Figure 6 we can see the variability that the pickup controller exhibits in terms of the location of the grasp on the gear, which leads to failures in attempting to insert this onto the base assembly. The issue comes from the tolerances of the fit as using an MP and a sequence of trajectory points does not compensate for any inaccuracies incurred during the previous stages of the process or manual positioning. Precise insertion is known to fail outside of a very small convergence basin when using MP controllers - we obtain similar (bad) performance similar to [2], [45].

TABLE I: Table of successful trials for different policies. MP - Motion Planning, DMP - Dynamic Motion Primitive, CNN - Convolutional Neural Network. The CNN policy has a maximum of 50 steps to reach the goal. The symbol * indicates policies terminated early due to safety concerns.

Control Method	Pick Up	Gear Move	Gear Insert	Full Task
MP	10/10	10/10	1/10	1/10
DMP	10/10	10/10	1/10	1/10
CNN	*	10/10	10/10	*
MP & CNN (Our)	10/10	10/10	10/10	**10/10**

As a baseline, we compare against optimally sequencing the MP and DMP control strategies, which can be seen under the "Full Task" performance. Due to the low performance on a part of the task, the overall success rate is limited.

Fig. 7: The execution of a neural network policy for inserting the gear on the peg.

In contrast, the natural variability of the grasp is part of the training set of the CNN model and successfully inserts the gear even with a high variance of initial locations (Figure. 7). As the visual CNN policy is not dependant on the absolute position of either the grasped location or the position of the base assembly, it performs corrective/feedback actions for the

policy to succeed. However, the CNN performance on the pickup task could not be evaluated, as the prescribed controller actions were jerky and violated safety constraints (pre-defined velocity and position limits).

This illustrates that the combination of **MP** for picking up the gear and moving it closer to the assembly and **CNN** to insert the gear, selected using our method allows for the full task to be successfully solved optimally 10 out of the 10 attempts. This shows the advantage of using a diverse set of controllers, allowing each one to be tuned to the domain of operation.

The Goal Score Model is trained on only 5 full task demonstrations. We empirically choose $n = 10$ for the n step MPC look ahead as our planning horizon. In the interests of reproducibility, more information about the sub-controllers and training routines is available on the website [3]. Figure 8 illustrates the Goal Score estimation for a previously unseen demonstration from camera streams with different viewpoints. The score for the different controllers can clearly be used to sequence the policies. This is shown by the fact that the score follows a monotonically increasing value with regards to the average score for the individual controller domain.

VI. Conclusion

We introduce a method for composing diverse policies with varied representations, including Motion Planning, Dynamic Motion Primitives and Convolutional Neural Networks. This allows for the solution of combinatorially complex and temporally extended tasks requiring multiple steps, without needing to predefine controller sequences or design high level state machines. We sequence tasks by using a Goal Scoring Model trained by expert demonstrations providing a weak supervisory signal. The goal scoring model provides a controller invariant prediction of progress towards a goal, which can be used with shared latent space across sub-controllers. This work has also introduced different methods that allow for a model-based or a model-free way to create a dynamics model, which can be used to analytically plan the next best option within a model predictive control framework.

References

[1] Ross A Knepper, Todd Layton, John Romanishin, and Daniela Rus. Ikeabot: An autonomous multi-robot coordinated furniture assembly system. In *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013.

[2] Garrett Thomas, Melissa Chien, Aviv Tamar, Juan Aparicio Ojea, and Pieter Abbeel. Learning robotic assembly from cad. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018.

[3] Matthew Thomas Mason. Manipulator grasping and pushing operations. pages 14–28, 1982.

[4] Robert R Burridge, Alfred A Rizzi, and Daniel E Koditschek. Sequential composition of dynamically dexterous robot behaviors. *The International Journal of Robotics Research*, 18(6):534–555, 1999.

[5] Russ Tedrake, Ian R. Manchester, Mark Tobenkin, and John W. Roberts. Lqr-trees: Feedback motion planning via sums-of-squares verification. *The International Journal of Robotics Research*, 29(8):1038–1052, 2010.

[6] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[7] Michael Burke, Svetlin Penkov, and Subramanian Ramamoorthy. From explanation to synthesis: Compositional program induction for learning from demonstration. *Robotics: Science and Systems (R:SS)*, June 2019.

[8] Matthew T. Mason. Mechanics and planning of manipulator pushing operations. *The International Journal of Robotics Research*, 5(3):53–71, 1986.

[9] Carlos E Garcia, David M Prett, and Manfred Morari. Model predictive control: theory and practicea survey. *Automatica*, 25(3):335–348, 1989.

[10] Tomas Lozano-Perez, Matthew T. Mason, and Russell H. Taylor. Automatic synthesis of fine-motion strategies for robots. *The International Journal of Robotics Research*, 3(1):3–24, 1984.

[11] Kevin M. Lynch and Matthew T. Mason. Dynamic nonprehensile manipulation: Controllability, planning, and experiments. *The International Journal of Robotics Research*, 18(1):64–92, 1999.

[12] Alberto Rodriguez, Matthew T Mason, and Steve Ferry. From caging to grasping. *The International Journal of Robotics Research*, 31(7):886–900, 2012.

[13] Stefan Schaal. Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In *Adaptive motion of animals and machines*, pages 261–280. Springer, 2006.

[14] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373, 2013.

[15] Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G. Bellemare. Deepmdp: Learning continuous latent space models for representation learning. In *ICML*, pages 2170–2179, 2019.

[16] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[17] Jessie Huang, Fa Wu, Doina Precup, and Yang Cai. Learning safe policies with expert guidance. In *Advances in Neural Information Processing Systems*, pages 9105–9114, 2018.

[18] Daniel Angelov, Yordan Hristov, and Subramanian Ramamoorthy. Using causal analysis to learn specifications from task demonstrations. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, AAMAS '19, pages 1341–1349, 2019.

[19] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469 – 483, 2009.

[20] J Zico Kolter, Pieter Abbeel, and Andrew Y Ng. Hierarchical apprenticeship learning with application to quadruped locomotion. In *Advances in Neural Information Processing Systems*, pages 769–776, 2008.

[21] Felipe Codevilla, Eder Santana, Antonio M López, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9329–9338, 2019.

[22] Andrew G Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13(1-2):41–77, 2003.

[23] Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *arXiv preprint arXiv:1806.06877*, 2018.

[24] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[25] Doina Precup. Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, page 80, 2000.

[26] Gary L Drescher. *Made-up minds: a constructivist approach to artificial intelligence*. PhD thesis, Massachusetts Institute of Technology, 1989.

[27] Richard E Fikes, Peter E Hart, and Nils J Nilsson. Learning and executing generalized robot plans. *Artificial intelligence*, 3:251–288, 1972.

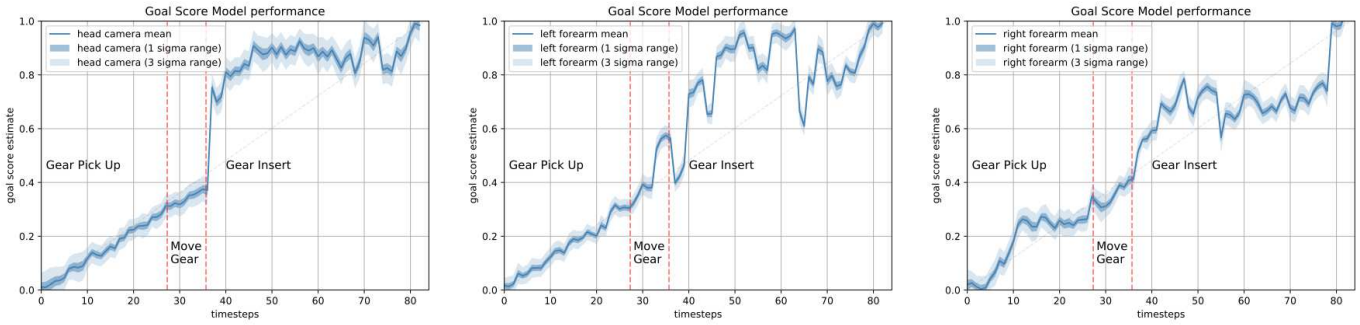
[28] Glenn A Iba. A heuristic approach to the discovery of macro-operators. *Machine Learning*, 3(4):285–317, 1989.

[29] Richard E Korf. Learning to solve problems by searching for macro-operators. Technical report, Carnegie-Mellon University, Pittsburgh, Dept of Computer Science, 1983.

[30] Petar Andonov, Anton Savchenko, Philipp Rumschinski, Stefan Streif, and Rolf Findeisen. Controller verification and parametrization subject to quantitative and qualitative requirements. *IFAC-PapersOnLine*, 48(8):1174–1179, 2015.

[31] Shromona Ghosh, Felix Berkenkamp, Gireeja Ranade, Shaz Qadeer, and Ashish Kapoor. Verifying controllers against adversarial examples with bayesian optimization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7306–7313. IEEE, 2018.

[3] https://sites.google.com/view/composingdiverse



(a) Head Camera (b) Left Forearm (c) Right Forearm

Fig. 8: The goal score metric calculated during the execution of a random trial. During the first two motion planning controllers, the model is monotonically increasing the goal metric. The stochasticity of the neural network policy leads to oscillating scores. Using different input streams, the prediction accuracy could be altered – the scene head camera does not see the fine details of the movement which the forearm cameras do, leading to a closer to goal score. The peaks in the forearm cameras are associated with states where the peg is extremely close to the gear hole, highlighting that proximity. Example snapshots from different views can be seen in Figure 9.

(a) Head Camera (b) Left Forearm (c) Right Forearm

Fig. 9: Snapshots of the input from different cameras on the PR2 robot demonstrate a moment, where the head camera cannot differentiate how well the task is performed, the left camera is optimistic from its perspective, while the right accurately evaluates the performance as sub-optimal, leading to the goal scoring network predicting a decreased value.

[32] Jean Harb, Pierre-Luc Bacon, Martin Klissarov, and Doina Precup. When waiting is not an option: Learning options with a deliberation cost. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[33] Danijar Hafner, Timothy P. Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *ICML*, pages 2555–2565, 2019.

[34] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 5048–5058, 2017.

[35] Maja J Mataric. Reward functions for accelerated learning. In *Machine Learning Proceedings 1994*, pages 181–189. Elsevier, 1994.

[36] Niko Sünderhauf, Oliver Brock, Walter Scheirer, Raia Hadsell, Dieter Fox, Jürgen Leitner, Ben Upcroft, Pieter Abbeel, Wolfram Burgard, Michael Milford, et al. The limits and potentials of deep learning for robotics. *The International Journal of Robotics Research*, 37(4-5):405–420, 2018.

[37] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1134–1141. IEEE, 2018.

[38] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, pages 2450–2462, 2018.

[39] Yilun Du and Karthic Narasimhan. Task-agnostic dynamics priors for deep reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1696–1705, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[40] Anna Harutyunyan, Peter Vrancx, Pierre-Luc Bacon, Doina Precup, and Ann Now. Learning with options that terminate off-policy, 2018.

[41] James J Kuffner and Steven M LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 995–1001. IEEE, 2000.

[42] Mel Vecerik, Oleg Sushkov, David Barker, Thomas Rothörl, Todd Hester, and Jon Scholz. A practical approach to insertion with variable socket position using deep reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 754–760. IEEE, 2019.

[43] Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. *Robotics: Science and Systems*, 2019.

[44] Tianhe Yu, Chelsea Finn, Annie Xie, Sudeep Dasari, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. OpenReview.net, 2018.

[45] Jianlan Luo, Eugen Solowjow, Chengtao Wen, Juan Aparicio Ojea, Alice M Agogino, Aviv Tamar, and Pieter Abbeel. Reinforcement learning on variable impedance controller for high-precision robotic assembly. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3080–3087. IEEE, 2019.